Exploring Speech-Enabled Dialogue with the Galaxy Communicator Infrastructure

Samuel Bayer
The MITRE Corporation
202 Burlington Rd.
Bedford, MA 01730
sam@mitre.org

Christine Doran
The MITRE Corporation
202 Burlington Rd.
Bedford, MA 01730
cdoran@mitre.org

Bryan George
The MITRE Corporation
11493 Sunset Hills Rd.
Reston, VA 20190
bgeorge@mitre.org

ABSTRACT

This demonstration will motivate some of the significant properties of the Galaxy Communicator Software Infrastructure and show how they support the goals of the DARPA Communicator program.

Keywords

Spoken dialogue, speech interfaces

1. INTRODUCTION

The DARPA Communicator program [1], now in its second fiscal year, is intended to push the boundaries of speech-enabled dialogue systems by enabling a freer interchange between human and machine. A crucial enabling technology for the DARPA Communicator program is the Galaxy Communicator software infrastructure (GCSI), which provides a common software platform for dialogue system development. This infrastructure was initially designed and constructed by MIT [2], and is now maintained and enhanced by the MITRE Corporation. This demonstration will motivate some of the significant properties of this infrastructure and show how they support the goals of the DARPA Communicator program.

2. HIGHLIGHTED PROPERTIES

The GCSI is a distributed hub-and-spoke infrastructure which allows the programmer to develop Communicator-compliant servers in C, C++, Java, Python, or Allegro Common Lisp. This system is based on message passing rather than CORBA- or RPC-style APIs. The hub in this infrastructure supports routing of messages consisting of key-value pairs, but also supports logging and rule-based scripting. Such an infrastructure has the following desirable properties:

- The scripting capabilities of the hub allow the programmer to weave together servers which may not otherwise have been intended to work together, by rerouting messages and their responses and transforming their keys.
- The scripting capabilities of the hub allow the programmer to insert simple tools and filters to convert data among formats.
- The scripting capabilities of the hub make it easy to modify the message flow of control in real time.
- The scripting capabilities of the hub and the simplicity of message passing make it simple to build up systems bit by bit.
- The standard infrastructure allows the Communicator program to develop platform- and programming-language-independent service standards for recognition, synthesis, and other better-understood resources.
- The standard infrastructure allows members of the Communicator program to contribute generally useful tools to other program participants.

This demonstration will illustrate a number of these properties.

3. DEMO CONFIGURATION AND CONTENT

By way of illustration, this demo will simulate a process of assembling a Communicator-compliant system, while at the same time exemplifying some of the more powerful aspects of the infrastructure. The demonstration has three phases, representing three successively more complex configuration steps. We use a graphical display of the Communicator hub to make it easy to see the behavior of this system.

As you can see in Figure 1, the hub is connected to eight servers:

- MITRE's Java Desktop Audio Server (JDAS)
- MIT SUMMIT recognizer, using MIT's Mercury travel domain language model
- CMU Sphinx recognizer, with a Communicator-compliant wrapper written by the University of Colorado Center for Spoken Language Research (CSLR), using CSLR's travel domain language model
- A string conversion server, for managing incompatibilities between recognizer output and synthesizer input
- CSLR's concatenative Phrase TTS synthesizer, using their travel domain voice

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **2001**	2. REPORT TYPE	3. DATES COVERED **00-00-2001 to 00-00-2001**

4. TITLE AND SUBTITLE **Exploring Speech-Enabled Dialogue with the Galaxy Communicator Infrastructure**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **The MITRE Corporation,202 Burlington Road,Bedford,MA,01730**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**3**	

- CMU/Edinburgh Festival synthesizer, with a Communicator-compliant wrapper written by CSLR, using CMU's travel domain language model for Festival's concatenative voice
- MIT TINA parser, using MIT's Mercury travel domain language model
- MIT Genesis paraphraser, using MIT's Mercury travel domain language model

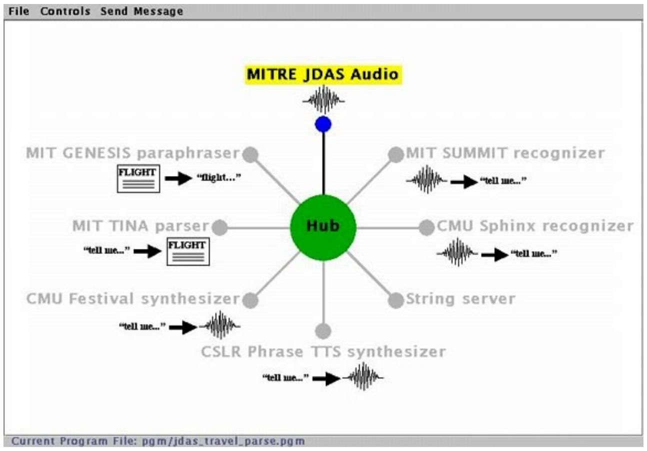

Figure 1: Initial demo configuration

We will use the flexibility of the GCSI, and the hub scripting language in particular, to change the path that messages follow among these servers.

3.1 Phase 1

In phase 1, we establish audio connectivity. JDAS is MITRE's contribution to the problem of reliable access to audio resources. It is based on JavaSound 1.0 (distributed with JDK 1.3), and supports barge-in. We show the capabilities of JDAS by having the system echo the speaker's input; we also demonstrate the barge-in capabilities of JDAS bye showing that the speaker can interrupt the playback with a new utterance/input. The goal in building JDAS is that anyone who has a desktop microphone and the Communicator infrastructure will be able to use this audio server to establish connectivity with any Communicator-compliant recognizer or synthesizer.

3.2 Changing the message path

The hub maintains a number of information states. The Communicator hub script which the developer writes can both access and update these information states, and we can invoke "programs" in the Communicator hub script by sending messages to the hub. This demonstration exploits this capability by using messages sent from the graphical display to change the path that messages follow, as illustrated in Figure 2. In phase 1, the hub script routed messages from JDAS back to JDAS (enabled by the message named "Echo"). In the next phase, we will change the path of messages from JDAS and send them to a speech recognizer.

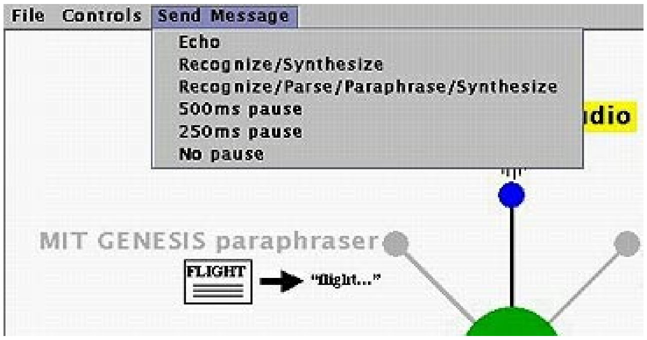

Figure 2: Modifying the hub information state

3.3 Phase 2

Now that we've established audio connectivity, we can add recognition and synthesis. In this configuration, we will route the output of the preferred recognizer to the preferred synthesizer. When we change the path through the hub script using the graphical display, the preferred servers are highlighted. Figure 3 shows that the initial configuration of phase 2 prefers SUMMIT and Festival.

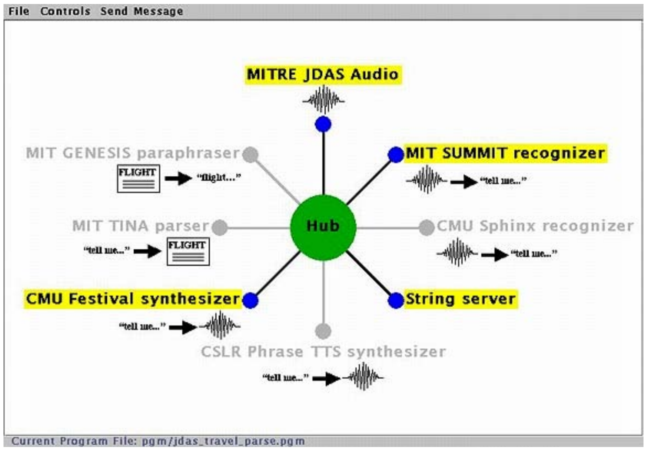

Figure 3: Initial recognition/synthesis configuration

The SUMMIT recognizer and the Festival synthesizer were not intended to work together; in fact, while there is a good deal of activity in the area of establishing data standards for various aspects of dialogue systems (cf. [3]), there are no programming-language-independent service definitions for speech. The hub scripting capability, however, allows these tools to be incorporated into the same configuration and to interact with each other. The remaining incompatibilities (for instance, the differences in markup between the recognizer output and the input the synthesizer expects) are addressed by the string server, which can intervene between the recognizer and synthesizer. So the GCSI makes it easy both to connect a variety of tools to the hub and make them interoperate, as well as to insert simple filters and processors to facilitate the interoperation.

In addition to being able to send general messages to the hub, the user can use the graphical display to send messages associated with particular servers. So we can change the preferred recognizer or synthesizer. (as shown in Figure 4), or change the Festival voice (as shown in Figure 5). All these messages are configurable from the hub script.

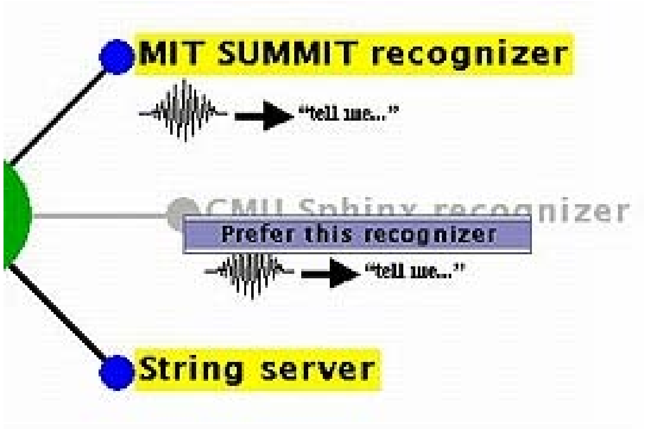

Figure 4: Preferring a recognizer

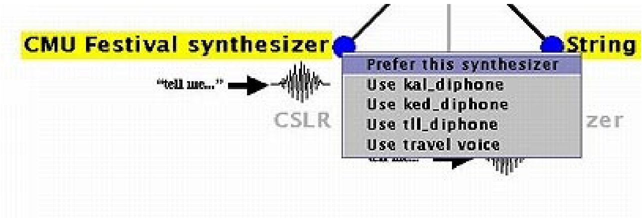

Figure 5: Changing the Festival voice

3.4 Phase 3

Now that we've established connectivity with recognition and synthesis, we can add parsing and generation (or, in this case, input paraphrase). Figure 6 illustrates the final configuration, after changing recognizer and synthesizer preferences. In this phase, the output of the recognizer is routed to the parser, which produces a structure which is then paraphrased and then sent to the synthesizer. So for instance, the user might say "I'd like to fly to Tacoma", and after parsing and paraphrase, the output from the synthesizer might be "A trip to Tacoma".

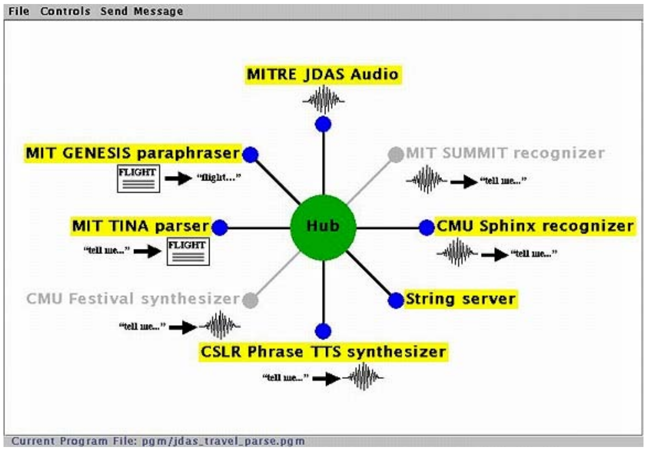

Figure 6: Adding parsing and paraphrase

4. CONCLUSION

The configuration at the end of phase 3 is obviously not a complete dialogue system; this configuration is missing context management and dialogue control, as well as an application backend, as illustrated by the remaining components in white in Figure 7. However, the purpose of the demonstration is to illustrate the ease of plug-and-play experiments within the GCSI, and the role of these capabilities to assemble and debug a complex Communicator interface. The GCSI is available under an open source license at http://fofoca.mitre.org/download.

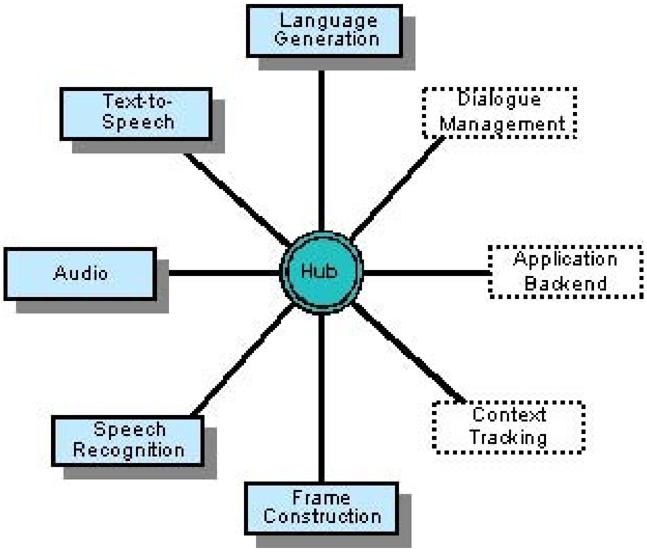

Figure 7: A sample full dialogue system configuration

5. ACKNOWLEDGMENTS

This work was funded by the DARPA Communicator program under contract number DAAB07-99-C201.

6. REFERENCES

[1] http://www.darpa.mil/ito/research/com/index.html.

[2] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. Galaxy-II: A Reference Architecture for Conversational System Development. Proc. ICSLP 98, Sydney, Australia, November 1998.

[3] "'Voice Browser' Activity." http://www.w3.org/Voice.